```
// C++ code to implement Fractional Knapsack Problem
#include <bits/stdc++.h>
using namespace std;
bool compare(vector<int>& a, vector<int>& b) {
    double a1 = (1.0 * a[0]) / a[1];
    double b1 = (1.0 * b[0]) / b[1];
    return a1 > b1;
}
double fractionalKnapsack(vector<int>& val, vector<int>& wt, int capacity) {
    int n = val.size();
     items[i][0] = value, items[i][1] = weight
   vector<vector<int>> items(n, vector<int>(2));
   for (int i = 0; i < n; i++) {
        items[i][0] = val[i];
        items[i][1] = wt[i];
    }
    sort(items.begin(), items.end(), compare);
   double res = 0.0;
    int currentCapacity = capacity;
    for (int i = 0; i < n; i++) {
         if (items[i][1] <= currentCapacity) {
            res += items[i][0];
            currentCapacity -= items[i][1];
        }
        else {
            res += (1.0 * items[i][0] / items[i][1]) * currentCapacity;
```

```
            // Knapsack is full
            break;
        }
    }
    return res;
}
int main() {
    vector<int> val = {60, 100, 120};
    vector<int> wt = {10, 20, 30};
    int capacity = 50;
     cout << fractionalKnapsack(val, wt, capacity) << endl;
    return 0;
}
```

📥 **Input:**
val = {60, 100, 120}; wt = {10, 20, 30}; capacity = 50;
📤 **Output:**
240

Time Complexity: O(n Log n)
Auxiliary Space: O(n)